

Linguissime

Spécifications

Version 1.0

4/04/2016

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| **4/04/2016** | 1.0 | Première version | Guillaume Danguin |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sommaire

1 Introduction [6](#__RefHeading___Toc254870962)

1.1 Contexte initiale, historique et vision [6](#__RefHeading___Toc254870963)

1.2 Mission [6](#__RefHeading___Toc254870964)

1.3 Objectifs [6](#__RefHeading___Toc254870965)

1.4 Glossaire [6](#__RefHeading___Toc254870966)

1.5 Documents de référence [6](#__RefHeading___Toc254870967)

2 Description générale [7](#__RefHeading___Toc254870968)

2.1 Acteurs [7](#__RefHeading___Toc254870969)

2.2 Cas d’utilisations [7](#__RefHeading___Toc254870970)

3 Spécifications fonctionnelles [9](#__RefHeading___Toc254870971)

3.1 Carte de navigation [9](#__RefHeading___Toc254870972)

3.2 Détails des cas d’utilisations [9](#__RefHeading___Toc254870973)

4 Spécifications non fonctionnelles [13](#__RefHeading___Toc254870974)

4.1 Environnement opérationnel [13](#__RefHeading___Toc254870975)

4.2 Contraintes de conception et d’implémentation [13](#__RefHeading___Toc254870976)

4.3 Documentation utilisateur [13](#__RefHeading___Toc254870977)

4.4 Exigences de performance [14](#__RefHeading___Toc254870978)

4.5 Exigences de sécurité [14](#__RefHeading___Toc254870979)

4.6 Attributs de qualité [14](#__RefHeading___Toc254870980)

Les spécifications sont à la base de l’accord entre le client et l’équipe réalisatrice sur ce que le système devra faire et ce qu’il ne devra pas faire.

Ce document est adapté à une description des fonctionnalités sous forme de cas d’utilisation.

Critères d’évaluations lors de l’avant-projet ou avant chaque début d’itération (\*) :

Spécifications fonctionnelles :

* Les acteurs (visiteur, administrateur, …) sont identifiés
* Les cas d’utilisations sont listés
* Les cas d’utilisations ont été détaillés + maquette (\*\*)
* La carte de navigation du logiciel est présente (\*)

Spécifications non-fonctionnelles

* Les contraintes de conception et d’implémentation ont été listées
* Les attributs de qualité ont été listés

(\*) Dans le cas d’un processus itératif, les spécifications détaillées (paragraphe 3) peuvent concerner l’itération courante ou plusieurs itérations

(\*\*) Dans le cas d’un processus itératif, les spécifications détaillées sont validés par le client à chaque début d’itération

# Introduction

## Contexte initial, historique et vision

Ayant fait le constat qu'il n'existe que très peu d'applications linguistiques gratuite et de très bonne qualité, nous souhaitons créer une plateforme d'apprentissage de langues centralisé autour d'exercices interactif.

## Mission

Linguissime est une application web accessible sur mobile, tablette et ordinateur permettant de créer facilement des exercices interactif en anglais et de les réviser.

## Objectifs

* Créer une fonctionnalité permettant de créer des exercices interactifs en moins de 5 minutes.
* Implémenter un moteur de recherche fournissant des résultats en dessous de 10 ms.
* Créer une fonctionnalité de gestion de d'images permettant de gérer au minimum 10 000 images.
* Mettre en production notre application sur une instance EC2 Amazon Web Service et faire du versionning de notre application.
* Créer notre propre framework CSS en ayant au moins 20 composants réutilisable nous permettant de concevoir le visuel de notre application et de ne pas dépendre des frameworks CSS open-source.
* Créer une fonctionnalité permettant d'envoyer des emails automatiques.
* Créer un système permettant de gagner des points et niveau pour un utilisateur.
* Créer un système de parrainage ou un utilisateur peut envoyer une invitation à un de ses contact en spécifiant l'email de son contact.
* Créer une un système permettant de créer des exercices interactifs à travers une configuration par défaut.
* Créer une fonctionnalité permettant à un utilisateur de visualiser son historique de révision.
* Créer un panel administrateur permettant de visualiser le nombre d'utilisateurs, d'exercices crée et de supprimer des exercices ou utilisateurs.
* Mis en place d'une stratégie SEO afin d'optimiser notre référencement.

## Glossaire

### Termes du domaine

### Termes techniques

* Instance : Serveur virtuel
* ORM : Couche d'abstraction de la base de données

## Documents de référence

Lister tous les documents, normes, rapports, sites Internet…, utiles à la compréhension du dossier et identifier leurs origines. Renvoyer aux annexes au besoin.

Exemples :

* Résultats d'interviews des parties prenantes
* Résultats d'ateliers et de sessions de définition des exigences
* Cahier des charges
* Exposé de la mission
* Règles métier
* Lois et réglementations
* Systèmes existants
* Modèles métier

# Description générale

## Acteurs

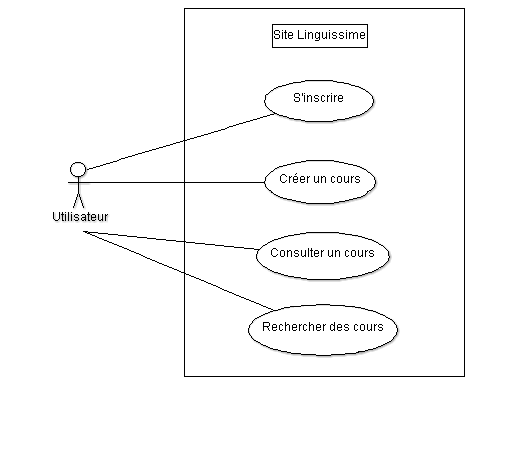
Présenter les différentes familles d’utilisateurs de la solution

|  |  |
| --- | --- |
| **Nom** | **Description** |
| Guillaume Danguin, Antoine Grandiere | Créateurs et développeurs du projet |
| Olivier Spinelli, Antoine Raquillet | Enseignants suiveurs |

## Cas d’utilisations

Lister les cas d’utilisations principales du produit, les détails seront donnés plus bas

### Diagramme de cas d’utilisation



### Listes des cas d’utilisations

Nommer et décrire brièvement les cas d’utilisation répondant aux exigences fonctionnelles de votre système.

Pour identifier les cas d'utilisation appropriés, commencez par réfléchir à ce que chaque acteur attend du système. Pour chaque acteur, humain ou non, posez-vous les questions suivantes :

* Du point de vue de l'acteur, quelles sont les principales tâches que le système doit exécuter ?
* L'acteur va-t-il créer, stocker, modifier, supprimer ou lire des données dans le système ?
* L'acteur sera-t-il amené à informer le système de changements extérieurs soudains ?
* L'acteur doit-il être avisé de l'occurrence de certains événements dans le système ?
* Une fois les cas d’utilisation identifiés, hiérarchisez-les en tenant compte des deux facteurs suivants :
* La priorité fonctionnelle
* Le risque technique
* L’objectif ensuite est de planifier en premier (premières itérations) la réalisation des cas d’utilisation ayant un risque élevé et une priorité haute.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom du cas d’utilisation** | **Acteur(s)** | **Priorité** | **Risques** |
|  |  |  |  |

# Spécifications fonctionnelles

## Carte de navigation

L'objectif de la carte de navigation est d'exprimer les chemins d'interface principaux dans le système. Ces chemins sont les chemins principaux à l'écran et ne reprennent pas la totalité des possibilités. Elle peut être considérée comme une carte routière de l'interface utilisateur.

Un diagramme de navigation peut être représenté grâce à un diagramme d’activité

## Détails des cas d’utilisations

Certains cas d’utilisation complexe avec de nombreuses extensions et/ou répétitions peuvent être illustrés par un diagramme d’activité.

### Nom du cas

Le nom du cas doit commencer par un verbe. Il correspond à l’objectif à atteindre.

#### Description :

Résumé succinct qui présente le scénario de base (succès) dans un paragraphe.

Peut être rédigé lors de la première étude de besoins et avant la rédaction détaillée du cas d’utilisation pour se faire une idée du sujet et de son périmètre.

#### Acteur principal :

Celui qui fait appel au système pour atteindre un but.

Un acteur n’est pas un utilisateur : une même personne peut jouer plusieurs rôles, plusieurs personnes peuvent jouer un même rôle.

Un acteur peut être un système, une machine.

#### Acteurs secondaires :

Optionnel.

Autres participants du cas d’utilisation.

#### Préconditions :

Ce qui doit toujours être vrai avant le début d’un scénario. En principe, une précondition implique que le scénario d’un autre cas d’utilisation s’est déroulé normalement (ex : « Le Caissier est identifié et authentifié »).

#### Postconditions (garantie en cas de succès) :

Ce qui doit être vrai lorsque le cas d’utilisation se termine avec succès, qu’il s’agisse du scénario principal ou d’un scénario alternatif.

#### Scénario nominal :

Il décrit le scénario type qui satisfait les intérêts de l’acteur principal et des acteurs secondaires. Il ne comprend pas de conditions ni de branchement. Les traitements conditionnels sont à reporter dans la section Extensions.

1. Le cas débute lorsque …

La première étape indique l’événement qui déclenche le scénario.

Exemple pour le cas d’utilisation Traiter une vente : « Le Client arrive à la caisse avec les articles qu’il souhaite acheter »

1. L’Utilisateur {verbe d’action} …
2. Le Système {verbe d’action} …

Le nom des acteurs doivent commencer par une majuscule.

Chaque étape d’actions est rédigée comme une action simple « dans laquelle le sujet est actif ». On peut la comparer à la description d’un match de football : « la pêrsonne1 envoie le ballon à la personne2 ; la personne 2 dribble ; la personne2 envoie le ballon à la personne3 ».

Le scénario est composé d’étapes, lesquelles sont de trois sortes :

1. Une interaction entre deux acteurs (le système étant considéré comme un acteur). Exemple : « Le client saisit son adresse », « Le Système affiche le prix de l’article »

2. Une étape de validation pour protéger les intérêts d’un acteur. Exemple : « Le Système valide le code secret ».

3. Un changement interne pour satisfaire aux intérêts d’un intervenant. Exemple : « Le Système enregistre la vente ».

Il peut être parfois utile d’utiliser des mots clés permettant d’exprimer la répétition

Exemple :

Pour chaque article trouvé

Le système affiche une image du produit

Le système affiche le prix du produit

…

1. Pour {expression d’itération}
2. Faire quelque chose
3. Faire quelque chose d’autre
4. Tant que {expression booléenne}
5. Faire quelque chose
6. Faire quelque chose d’autre

#### Extensions:

Les extensions permettent de décrire les autres scénarios ou branchement possibles, tant en cas de succès qu’en cas d’échec.

3a . Si {*Condition d’extension de l’étape 3 du scénario nominal– à compéter}*  alors:

Comme pour la première étape du scénario nominal, la condition d’extension indique l’événement qui déclenche le scénario.

Exemple : « Si le code de l’article est invalide (non trouvé par le Système) alors »



Description des étapes permettant de prendre en charge la condition d’extension

#### Interface utilisateur:

Représenter les écrans de l’application. Il convient de rappeler que la maquette n’est pas le produit final !

Vous pouvez pour cela dessiner les écrans à la main ou utiliser un outil spécifique de création d’interface utilisateur (exemple : <http://www.mockupscreens.com/>, http://c2.com/cgi/wiki?GuiPrototypingTools)

#### Fréquence :

Optionnel.

Fréquence moyenne estimé d’apparition de ce cas d’utilisation par jours

#### Spécifications particulières :

Optionnel.

Besoin non fonctionnel, attribut de qualité ou contrainte se rapportant spécifiquement à ce cas d’utilisation. Il peut s’agir de questions de performance, de fiabilité ou d’ergonomie. Exemple : « Interface utilisateur à écran tactile sur grand écran plat. Le texte doit être visible à un mètre. ».

#### Questions en suspens:

Optionnel.

# Spécifications non fonctionnelles

## Environnement opérationnel

Décrire l’environnement dans lequel le produit devra fonctionner.

Pour une application informatique par exemple : le matériel, le système d’exploitation et sa version, la localisation géographique des utilisateurs, les serveurs, les bases de données, etc.

Lister les autres produits, systèmes, applications informatiques, avec lesquels le produit devra coexister pacifiquement

## Contraintes de conception et d’implémentation

Décrire tous les facteurs qui vont restreindre la liberté des concepteurs, donner la raison de chaque contrainte

Types de contraintes :

* Technologies, outils, langages de programmation, bases de données, spécifiques imposées ou interdites
* Restriction sur la version du système d’exploitation ou du navigateur Web
* Règles ou conventions imposées dans l’entreprise : structure de la documentation technique destinée à une future maintenance par exemple
* Compatibilité avec des produits plus anciens
* Limitations imposées par les règles du métier
* Limitations liées au matériel disponible : temps de réponse, taille mémoire, vitesse du processeur, taille, poids, matériaux, coûts
* Conventions liées à l’interface utilisateur, lors d’une extension par exemple
* Format standards d’échange d’information : XML par exemple

## Documentation utilisateur

Lister les composants de la documentation à destination des utilisateurs qui sera livrée avec le produit

Ceci comprend : les manuels utilisateurs, l’aide en ligne et les tutoriaux

Identifier pour chaque document le format, les standards à respecter, les outils à utiliser

## Attributs de qualité

Indiquer ici tous les autres attributs de qualité importants pour l’utilisateur et les développeurs. Ces caractéristiques doivent être spécifiques, quantitatives et vérifiables

Indiquer les priorités des divers attributs. Par exemple : « la facilité d’apprentissage est plus importante que la facilité d’installation »

Les attributs de qualité importants pour les utilisateurs sont :

* Performances spécifiques nécessaires à la bonne marche du produit. Donner les raisons pour faciliter le travail des concepteurs. Quantifier au maximum
* Disponibilité : up-time
* Efficience : bonne utilisation des ressources
* Flexibilité : extension, augmentation, réduction, etc.
* Intégrité : gestion des accès, protection des données, etc.
* Interopérabilité : échange de données et de services avec d’autres systèmes
* Fiabilité : probabilité d’exécution sans erreur
* Robustesse : tolérance aux fautes (Tankness en argot de développeur US)
* Utilisabilité : facilité d’utilisation (user friendly)
* Sécurité : toute exigence concernant la sécurité, l’intégrité, le respect de la vie privée. Considérer le produit, l’usage du produit, les données que le produit utilise ou crée.

Les attributs de qualité importants pour les développeurs sont :

* Maintenabilité : facilité de correction ou de modification
* Portabilité : facilité de migration d’un environnement à un autre
* Ré-utilisabilité : notion de composant
* Testabilité : facilité de test.